FIRST ORDER MOTION MODEL FOR IMAGE ANIMATION AND DEEP FAKE DETECTION

USING DEEP LEARNING

Mrs. Banu Priya M
Department of Computer Science, Sri Shakthi Institute of Engineering and Technology
Coimbatore, India
mbanu26@gmail.com

Mr. Jhosiah Felips Daniel
Department of Computer Science, Sri Shakthi Institute of Engineering and Technology
Coimbatore, India
dfelips@gmail.com

2022 International Conference on Computer Communication and Informatics (ICCCI) | 978-1-6654-8035-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICCCI54379.2022.9740969

Abstract— **Image animation involves animating an image using a video. In our case, the image contains the face of a person. This face is animated (creation of facial expression) based on the motion of the video frame (may contain a person talking). Current methods involve animating an image with respect to different videos. Our findings show that these methods lack certain features – such as audio capabilities (mimicking the voice of the person in the video). We propose that the chosen image be trained with a variety of videos which contain items of the same classification – say facial features of humans such as eyes, lips, nose etc. and along with the audio output of these items. Self-supervised formulation approach is adapted for the training purpose. Mapping between the motion of the item in the video and the image can be achieved with the help of generator network models. The result is a generation of a model that can mimic any video of the specified classification with the desired audio output.**

Keywords: ***Deep Faking; Video; Animation; Deep Learning;***

I. INTRODUCTION

The principle of our proposal relies on the deep faking of videos. Deep faking involves replacing someone or something in an existing video with another version of that someone or something – literally manipulating the video according to one's own needs. This paper discusses the various methods used for Deep-Faking. Our proposed approach focuses on one Method - The Monkey-Net method to deep fake in terms of animating a source image S with respect to the motions from a driving video D.

II. RELATED WORKS

(1) explains how deep videos can be generated based on Temporal Generative Adversarial Nets (TGAN). TGANS are used along with a discriminator to differentiate between videos generated by a video generator and those coming from the actual source. Firstly, a set of variables are generated by the Temporal generator. These variables are then transformed into a video containing T number of frames by the image generator. Comparison of the frames generated by the image generator with the actual video frames are done by the discriminator. The output is a detection of deep-faking in videos. Hence TGANS are used not only to generate deep-fakes but also in detection of the same. Figure 1 explains this:

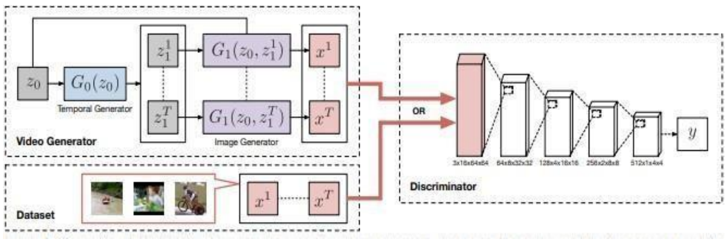

Figure 1. Illustration of the TGAN

(1) is also related to the deep-faking of video frames but in a different manner. Here, the background and foreground of the video frames are analyzed. For example, a new born baby laying on a cradle in a hospital. The foreground in this example is the baby and the background is the cradle. The video is processed such that the baby object is disentangled/extracted from its cradle resulting in the cradle as the output. A second video containing a different baby is taken and processed in the same way but in this case only the baby is taken from the output. This baby is then placed in the background extracted from the first video – resulting in a perfectly deep faked video. The following figure illustrates this:

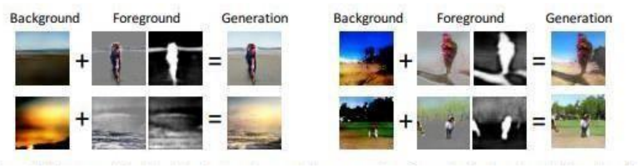

Figure 2. The Streams concept

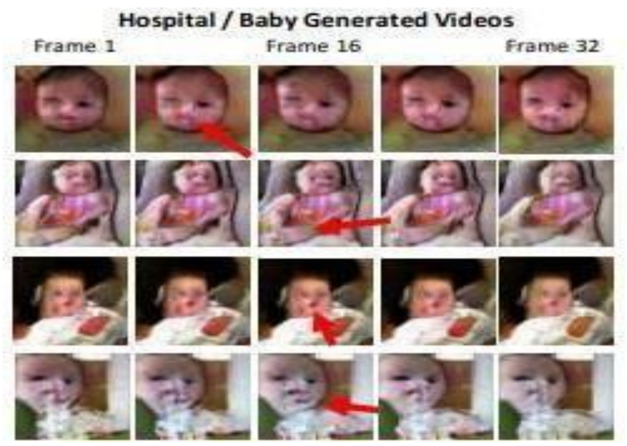

Figure 3. Images generated by a Generator Network

To achieve the above, a Generator Network is used. One of the proposed architectures for the Generator Network is the Two Stream Architecture. This architecture is useful for videos that involve a stationary background and mobile foreground – a good example would be a cradle holding a restless crying baby. The background is denoted by b(z) and foreground is denoted by f(z). m(z) denotes the generated masked video containing the background and foreground. This results in the following formula:

(3) explains another way of deep faking videos. Different people smile in different ways. A variety of facial expressions – such as the ways in which the cheek or upper lip is raised are seen across different people when they smile. The idea here is to generate different smiles for a single person.

The approach used here is known as the Conditional Multi-Mode Generation. The architecture comprises of three blocks:

- The conditional recurrent landmark generator
- The multi-mode recurrent landmark generator
- The landmark sequence to video translation module

Firstly, a face is given as the input to the conditional recurrent landmark generator – this encodes the given image into a compact representation and also gives an output sequence in relation to the opted facial expression class. The multi-mode recurrent landmark generator receives this

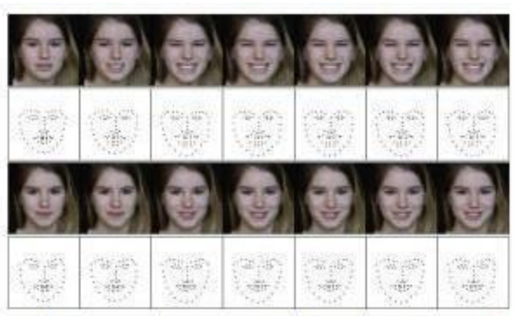

Figure 4. Variances in facial expressions

sequence and generates K number of sequences of the same facial expression class.

Finally, the video translation module fetches the landmark sequences along with the original face image to produce different facial expression videos as the output – causing one person to facially express themselves in multiple ways. The following image shows this architecture:

(3) A video obviously has to have some movement/motion in it. If not, then it isn't a video. For example, take a ballerina performing some moves on a stage. The deep fake concept is used here in making the ballerina perform moves that she never really performed. The approach describes here uses an image latent space to achieve this. To start off, this space is divided into two subspaces – the content subspace and the motion subspace. Content subspaces consist of points while motion subspaces consist of trajectories. By sampling these points and trajectories, different movements/motion of the same object can be achieved. The following figure shows such examples:

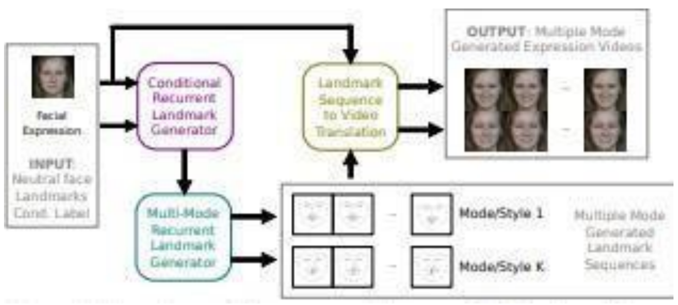
Facial Expression
INPUT: Neutral face Landmarks Cond. Label
Conditional Recurrent Landmark Generator
Landmark Sequence to Video Translation
OUTPUT: Multiple Mode Generated Expression Videos
Multi-Mode Recurrent Landmark Generator
Mode/Style 1
Mode/Style K
Multiple Mode Generated Landmark Sequences

Figure 5. Generatoion of facial expression videos

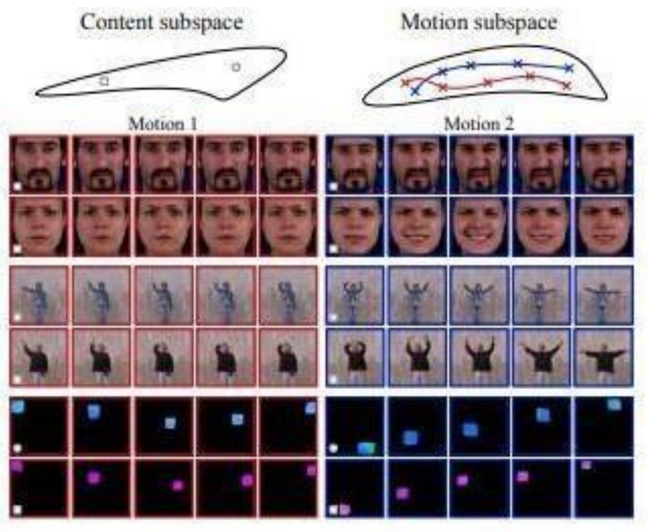

Figure 6. Concept of MoCoGAN

The common problem faced by the methods described is the process of rendering the final face output. The structure/geometry of the real-world face is described in terms of triangles and quad meshes. Ways in which the face/skin interacts with light is complex. All these factors have to be carefully considered in order to create a close to accurate representation of the face. As an example, let us look at how the facial skin surface details such as wrinkles are maintained in the digitally rendered face. (16) describes a method called Fine-Scale Detail Estimation which makes use of shape-from shading for this purpose. Shading cues are added to the input face image. The cues are then used in the refinement of the surface geometry. The con of this method is that it exhibits high computational complexity. An alternative to the approach is to render these fine details from high quality scans of facial images.

III. Works related to image animation

(16) talks about face tracking and reconstruction. The human face is something very unique. It is a medium through which identity, emotions and intentions can be expressed. Computerized tools are used in the generation of digital faces from the analysis of these real-world faces. Setups are important for capturing a real-world face. Some of these include:

- Setups to capture multiple-views
- Standard setups to capture multiple-views

Multi view setups use cameras in an indoor environment with controlled lighting conditions. Some of the other similar setups use markers placed on a face to capture it along with a light source from a device such as a projector. Standard multi view setups consist of the subject surrounded by a pairwise set of stereo cameras. The output from each pair of cameras is used in the construction of the face structure/geometry visible /captured by that particular set of cameras – a process called triangulation is used. These individual outputs are then used in the generation of the final face.

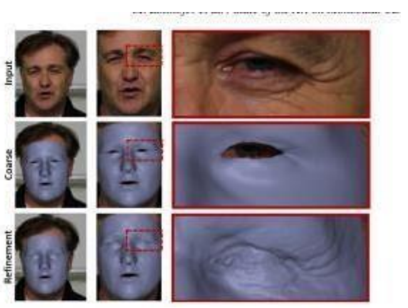

Figure 7. Facial performance capture

(21) describes how to generate a video from a given input. A learned video synthesis model is used for this purpose. Unlike the previous approach described, this method allows realistic videos to be generated without any specification of object geometry. Dynamics and lighting conditions. Some of the relevant applications (related to our study) for which this method is put to use are:

- Sketch to video synthesis for face swapping
- Pose to video synthesis for human motion transfer

Sketch to video synthesis for face swapping – used in the conversion of sketch sequences to videos.

Pose to video synthesis for human motion transfer – used in conversion of human motion/movements/poses to output videos which can contain new types of poses. The motion can also be transferred to another human.

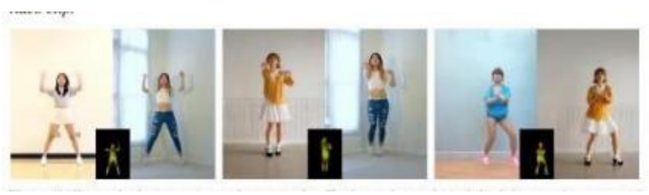
Figure 8. Concept of dance-poses

The methods described above require prior information about the object and are time consuming in the sense that long duration videos labeled with semantic details for the object/person under consideration have to be captured and maintained. However there exist some methods which do not require any prior information about the object/person. [23] describes how the expression of posture of a given face can be controlled using another face with the help of a neural network model (can be used with deep fake). [23] calls the architecture used to achieve this as the X2Face architecture. Two inputs are given to the X2Face. The first is a source video containing individual/multiple frames of the same face. The second is a driving frame containing individual/multiple frames from another face or the same face. The output of the X2Face contains a video with the facial features and looks of the face similar to the source video but the expressions and poses similar to those in the driver video. The following figure illustrates this:

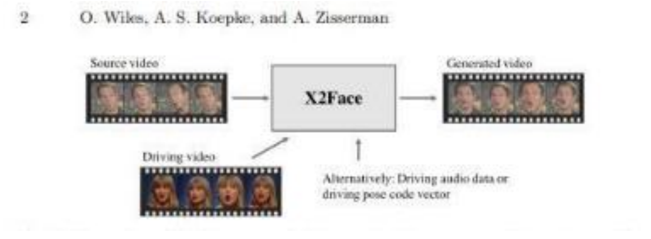
2 O. Wiles, A. S. Koepke, and A. Zisserman
Source video
X2Face
Generated video
Driving video
Alternatively: Driving audio data or driving pose code vector

Figure 8. The X2F Model

(24) describes how a given input image can be animated using the motion sequences from the video (which too is given as an input). Similar to the above method, a driving input i.e. the source video with motion sequences and an image source is fed as input. A generated video (produced as output) shows that the subject in the source image is animated with respect to the motion sequences of the subject in the driving video. The following figure illustrates this:

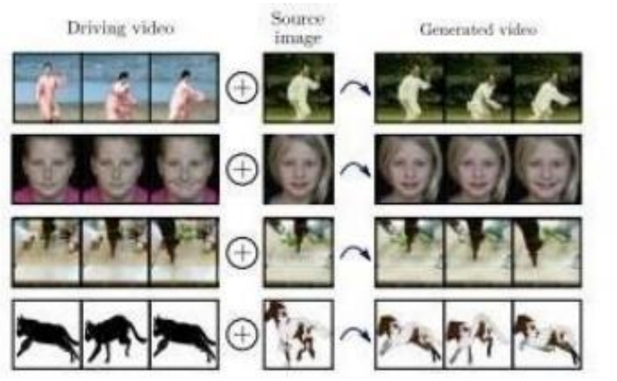

Figure 9. Generating output from driver video and source image

IV. METHODS

A. The Monkey-Net Architecture

An image source (S) and a driving video (D) can be used. S is to be animated based on the motions from D. The Monkey-Net architecture can be used for this purpose. Monkey-Net follows a self-supervised strategy. The Monkey-Net architecture is shown below:

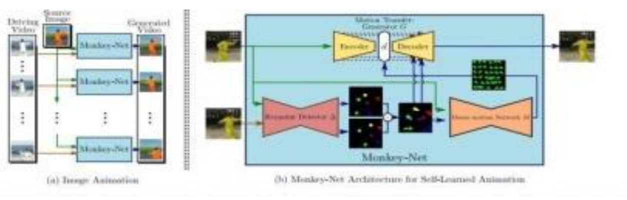

Figure 10. Motion tansfer framework

The sole purpose of the Monkey-Net architecture is to animate a source video based on motion from driving video. The architecture consists of three modules:

- The Keypoint Detector (Δ)
- The Dense Motion prediction network (M)
- The Motion Transfer Network (G)

The Keypoint Detector is fed with two inputs: An image source (S) along with a frame from a Driver video (D). Sparse keypoints are then extracted from these inputs and given to the Dense Motion predicting network which does the process of converting its inputs into heatmaps of motions. Finally, the Motion Transferring Network combines the heatmaps and image source to output a target frame – i.e., an frame with the image animated using the motions present in D.

Advantages of the Monkey-Net Architecture are as follows:

- Only a dataset with objects of interest is required
- No specifications such as labels or keypoint annotations are needed
- Direct supervision is not needed i.e. - the process is completely self-supervised
- Only two inputs: S and D need to given for training

Logic:

x , x' $\in X$ are two frames with size $H \times W$ respectively - they are extracted from the same video. A $H \times W$ lattice is represented by U. Δ together with G is used as follows: G should be able to recompute x' from the keypoint locations $\Delta(x) \in U$, $\Delta(x') \in U$, and x. As a result, the motion between x and x' is modeled.

For larger motions, factors that affect motion and object geometries need to be taken into consideration. A third network M is used in calculating the flow of optics between F $\in$ R and H×W×2 between x' and x from Δ(x), $\Delta(x')$ and x. This does the following:

Firstly, it forces Δ to make predictions about the keypoint locations that capture both object structure/geometry and motion. These key points locations must originate from places (in the object) where there is a high probability of movement. As an example, we can take the human body. Here the places where movements are most likely to occur would be at the hands and feet – hence these points are a good source for keypoint locations.

Secondly, to prevent misalignment of pixels between S and D, the flow of optics computed by M is utilized by G in generating the final output frame.

B. Monkey-Net Architecture Variant

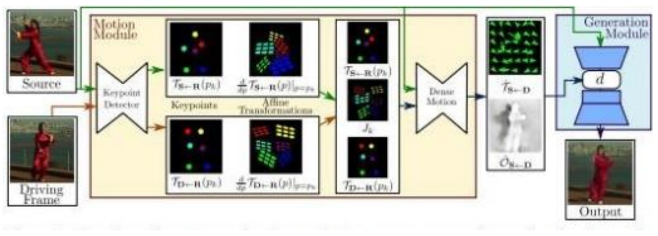

Figure 11. The Monkey-Net Architecture

This architecture is similar to Monkey-Net Architecture. It resembles in the following ways:
• Unsupervised learning
• Two inputs – S&D
• Dataset with objects belonging to a specific category

Unlike the original Monkey-Net Architecture, this architecture consists of two major modules:

- The Module for Motion Estimation
- The Module for Image Generation.

Logic

- R denotes an abstract frame
- S denotes the source frame
- D denotes the driving video frame
- Both S and D belong to R
- H and W represent height and width respectively
- T represents transformation

Consider a frame from the driving video D. Frame D ∈ R 3×H×W has a dimension of H × W. A source frame S exists. The source frame S ∈ R has a dimension 3×H×W. There exists a motion field between D and S. The motion estimation module is used in the prediction of this field.

The field of motion is described by the following function: T S←D: [R 2 → R 2]

What this means is that, each pixel location in D is being mapped/transformed to its equivalent location in S. The arrow is pointing from D to S. Hence this is referred to as an optics that flows backwards. Backward optical flow is used for its efficiency.

Two transformations are considered here:
• R to S (T S←R)
• R to D (T D←R)

The Motion Estimation Module:
The Motion Module consists of the Keypoint Detector and a Dense Motion Network.

a). Keypoint Detector

An encoder-decoder network is used in predicting keypoint locations in D and S. As explained in the original Monkey-Net architecture, these keypoints are used in representing motion. Keypoint trajectories from the D can be used to animate/move key points present in S. Local affine transformation (as we shall see shortly) is used in this motion generation process for each keypoint and its neighbors. In short, the keypoint locations and parameters required for motion generation are all computed by the keypoint detector present in the motion module.

b). Dense Motion Network

Computations from the Keypoint Detector are used in obtaining the dense motion field: Tˆ S←D This motion field (that is dense) is used in aligning the feature maps from S with the object’ s pose in D.

c). *The Image Generation Module*

The final step is the production of an image from S which is animated based on the motion from D. For this purpose, a generator network G is used.

d). *Describing motion using local affine transformations*

As mentioned earlier, the Motion Estimation Module makes use of backward optical flow for mapping D locations to S locations: T S←D

To carry out the above transformation, an abstract reference frame R is necessary. It is a two-set transformation process:
First from T R←D (D to R)
Then from T S←R (R to S)

For any given frame X, the transformation in that neighbourhood of motion key points obtained would be: T X ←R Transformations are expanded using the Taylor expansion method. For any given transformation say, T X←R, its expansion in terms of K key points would be as follows: p1, p2,........... , pK These terms are the coordinates of the key points in R. The coordinates for points in R are represented by p while those in frames X, S or D are represented by Z.

The following can be obtained:

T X←R(p) = T X←R(pk) + (d/dp * T X←R(p) | p=pk) (p − pk) + o (||p − ||pk) - (1)

Here, the motion function T X←R is denoted by values present in each keypoint pk. The Jacobians for the same is computed in each pk location as follows:
T X←R(p) - {{T X←R(p1), d /dp * T X←R(p) | p=p1} ⋯. {{T X←R(pk), d/dp * T X←R(p) | p=pk}} - (2)

To compute T R←X = T −1 X←R, T X←R is taken to be locally bijective in the neighbourhood of each keypoint. Let zK be the pixel location in D. each zK corresponds to keypoint location pK in R. To estimate T S←D the following computations are made:

- Estimate T R←D close to zK in D. Thus, pK= T R←D(zK)
- Estimate T S←R close to pk in R.

Based on these two estimations, the mapping from D to S can be done as follows: T S←D = T S←R ◦ T R←D = T S←R ◦ T −1 D←R – (4)

C. Occlusion-aware Image Generation

Occlusion in image means that some parts of the image produced are not perfectly rendered due to mismatch of pixels. S is not perfectly aligned in terms of pixels with the image (Dˆ) that is to be produced. [20] describes a feature warping strategy that could be used for this purpose. Here, a single person is used in training the model. The result is renderings of this person with new body postures or movements that were not discovered during the training process. The following figure shows this:

Figure 12. Training models to mimic human behaviour

The remedy to this problem of occlusion is to obtain feature maps. Backward optical flow is not enough to deal with the presence of occlusions in S. Hence inpainting i.e., filling in missed out details should be implemented in properly rendering the final output image. The keypoint representations obtained are put to use by adding a channel to the final layers of the dense motion network - this is for estimation of the occlusion mask. The feature map computed is then added to layers of the generation module to produce the final image.

D. *Motion Transfer from D to S – the final stage*

The sole purpose of this method is to animate S with respect to the motions in D. In detail, a source frame S1 has to be animated with respect to the driving frame video D1 DT. Each frame in D is processed to obtain S. To transfer the motion from D to S, the relative motion between D1 and DT is transferred to S. The advantage of transferring motion over relevant coordinates is that it allows for the transfer of relevant motion and helps preserve the overall structure/geometry of the object in interest.

V. CONCLUSION

Not all deep fakes are of threat. However, since the deep faking technology makes it effortless to create realistic media, some users are misusing it to perform malicious attacks. These attacks are targeting individuals - causing psychological, monetary, and physical problems. In this survey, the focus was on creation and detection of deep fake images and videos. An in-depth review in relation to how deep fake technologies were created, and what is being done to detect these deep fakes were given in this paper. We believe this information will be helpful to understand and prevent threatening deep fakes. Our future research focuses on one Method - The Monkey-Net method to deep fake in terms of animating a source image S with respect to the motions from a driving video D.

REFERENCES

[1] 1] Temporal Generative Adversarial Nets with Singular Value Clipping - Masaki Saito, Eiichi Matsumoto, Shunta Saito [Submitted on 21 Nov 2016 (v1), last revised 18 Aug 2017 (this version, v3)] https://arxiv.org/abs/1611.0662

[2] [3] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In NIPS, 2016.

[3] [4] Wei Wang, Xavier Alameda-Pineda, Dan Xu, Pascal Fua, Elisa Ricci, and Nicu Sebe. Every smile is unique: Landmark-guided diverse smile generation. In CVPR, 2018.

[4] [5] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In CVPR, 2018.

[5] [6] Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H Campbell, and Sergey Levine. Stochastic variational video prediction. In ICLR, 2017.

[6] [7] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In NIPS, 2016.

[7] [8] JoostVanAmersfoort, AnithaKannan,Marc'Aurelio Ranzato, Arthur Szlam, Du Tran, and Soumith Chintala. Transformation-based models of video sequences. arXiv preprint arXiv:1701.08435, 2017.

[8] [9] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In NIPS, 2016.

[9] [9] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action- conditional video prediction using deep networks in atari games. In NIPS, 2015.

[10] [10] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In ICML, 2015.

[11] [11] Joost Van Amersfoort, Anitha Kannan, Marc'Aurelio Ranzato, Arthur Szlam, Du Tran, and Soumith Chintala. Transformation-based models of video sequences. arXiv preprint arXiv:1701.08435, 2017.

[12] [12] Long Zhao, Xi Peng, Yu Tian, Mubbasir Kapadia, and Dimitris Metaxas. Learning to forecast and refine residual motion for image-to-video generation. In ECCV, 2018.

[13] [13] Chen Cao, Qiming Hou, and Kun Zhou. Displaced dynamic expression regression for real-time facial tracking and animation. TOG, 2014.

[14] [14] Zhenglin Geng, Chen Cao, and Sergey Tulyakov. 3d guided fine-grained face manipulation. In CVPR, 2019.

[15] [15] Justus Thies, Michael Zollhofer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Face2face: Real-time face capture and reenactment of rgb videos. In CVPR, 2016

[16] [16] Michael Zollhöfer, Justus Thies, Pablo Garrido, Derek Bradley, Thabo Beeler, Patrick Pérez, Marc Stamminger, Matthias Nießner, and Christian Theobalt. State of the art on monocular 3d face reconstruction, tracking, and applications. In Computer Graphics Forum, 2018.

[17] [17] Egor Zakharov, Aliaksandra Shysheya, Egor Burkov, and Victor Lempitsky. Few-shot adversarial learning of realistic neural talking head models. In ICCV, 2019.

[18] [18] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. In ECCV, 2018.

[19] [19] Aayush Bansal, Shugao Ma, Deva Ramanan, and Yaser Sheikh. Recycle-gan: Unsupervised video retargeting. In ECCV, 2018.

[20] Aliaksandra Shysheya, Egor Zakharov, Kara-Ali Aliev, Renat Bashirov, Egor Burkov, Karim Iskakov, Aleksei Ivakhnenko, Yury Malkov, Igor Pasechnik, Dmitry Ulyanov, Alexander Vakhitov, and Victor Lempitsky. Textured neural avatars. In CVPR, June 2019.

[20] [21] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. In NIPS, 2018.

[21] [22] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In CVPR, 2017.

[22] [23] Olivia Wiles, Sophia Koepke, and Andrew Zisserman. X2face: A network for controlling face generation using images, audio, and pose codes. In ECCV, 2018.

[23] [24] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. Animating arbitrary objects via deep motion transfer. In CVPR, 2019.

[24] Artur Grigorev, Artem Sevastopolsky, Alexander Vakhitov, and Victor Lempitsky. Coordinate- based texture inpainting for pose-guided image generation. In CVPR, 20